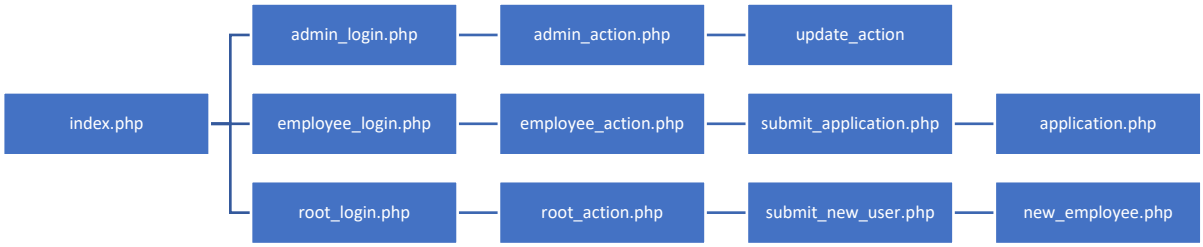
Leave Submission/Management System Documentation

By Angelos Gketsis

Due to a lack of knowledge in deploying servers for applications the whole application was made using the MySQL type usbserver, with PHP 7.1 and Apache 2.4.29.



Firstly, in the above diagram the total of the applications files can be seen. Giving us a clear view of which file/page leads to which.

In the next few pages each file and its use will be explained in detail, later we will explain the usage of the database that drives the application and the use of each of its tables.

File Analysis

index.php

Starting of with the door to our application the index file, inside this file we have three html forms in the way of buttons where the user decides in what way they want to log in. Depending on the choice the user will be lead either into admin_login.php or employee_login.php or root_login.php.

admin_login.php / employee_login.php / root_login.php

All these three files have the same function. They get the username and password info from the user and by way of POST send the information and the user to the appropriate page.

admin_action.php

Here a user with admin access can view all the pending submissions by ascending order. First off, we get the submitted username and password and pass them to the users session as seen below so we can keep the posted data and the user does not have to relog in.

```
if(!is_null($_POST["usern"]))
    {
        $usern = $_POST["usern"];
        $_SESSION["usern"] = $usern;
    }
    else
    {
        $usern = $_SESSION["usern"];
    }
    if(!is_null($_POST["pass"]))
    {
        $pass = $_POST["pass"];
        $_SESSION["pass"] = $pass;
    }
    else
    {
        $pass = $_SESSION["pass"];
    }
```

The process can be seen above, we check if the posted values are NULL if they aren't we pass them to our php program but if they are NULL, we get them from the users session.

We then get data from our database table named supervisor using a SELECT query based on the username submitted if no account exists with the given credentials the user gets a *"Account does not exist!"* message. If there is an existing account with the given credentials then we check if the users submitted password matches the password, if the passwords are not a match then the user gets an *"Invalid password"* and we leave the **$valid** variable False message otherwise we set the **$valid** variable True our php program carries on.

```
$sql1 = "SELECT `id`, passw FROM supervisor WHERE username LIKE '".$usern."'";
    $result  = $mysqli->query($sql1);

    $id;
    $password;
    if ($result->num_rows > 0)
    {
        while($row = $result->fetch_assoc())
        {
            $id = $row["id"];
```

```
            $password = $row["passw"];
        }
    }
    else
    {
        echo "Account does not exist! <br>";
    }

    $valid = false;

    if($pass == $password)
    {
        echo "Valid password <br>";
        $valid = True;
    }
    else
    {
        echo "Invalid password <br>";
    }

    $_SESSION["admin_status"] = $valid;
```

Furthermore, we set the session variable **admin_status** to match our **$valid** variable so even if the user gets to other pages meant for admins all they get is a *"Invalid Permissions"* message. Then if all the permissions and credentials are as they should the user gets a list of all the pending applications by ascending order, we get that using the following SELECT query.

```
$sql2 = "SELECT `id`, application_id, date_submitted, `startd`, `endd`,
Reason, `statuss` FROM `applications` WHERE statuss LIKE 'pending' ORDER BY
date_submitted ";
```

Below the list the user gets a form where they can input the application id of the application they want to update and the change to the application from two choices *"Approved"* or *"Declined"*, then the user presses the submit button and is taken to the update_application.php page.

update_application.php

This page is simple we take the posted data and our session data check if the user has privileges to make the UPDATE query to our database table applications based on the **admin_status** variable. If the user does not have permission to submit an UPDATE query the user gets an *"Invalid Permissions"* if the user has permissions, then their query is executed if all goes well with the update query *"Record Updated successfully"* if for some reason the update query can not be executed correctly then the user gets the appropriate message.

The query we use is

```
$sql = "UPDATE `applications` SET `statuss`= '$up_status' WHERE application_id
LIKE '$app_id' ";
```

employee_action.php

In this page a user with employee privileges can view past applications, the validation process of the users credentials is the same as showcased in the **admin_action.php** with the core difference being

that we get data from the database table employee also the listing process of old application is the same as the listing process of pending application on admin_action.php. Another key difference is that when we pass the user privileges in the users session in the form of the variable **employee_privilages** we also pass in the session the employee id of the user to be user in later leave applications. Above the list of past applications there is a button for the user to press named *"Submit Request"* when pressing this button, the user it taken to **submit_application.php.**

submit_application.php

Here an employee can submit the Start Date and End Date and the reason of their leave application using a form which later posts the submitted data to application.php. All the above fields are simple text based inputs.

application.php

In this file the application process is quite simple and has more than enough similarities to the **update_application.php** file centering on the validation process, a core difference being we do not update any applications on our database table applications we use an INSERT INTO query inserting the data submitted to the table. If the user does not have permission to submit an INSERT query the user gets an *"Invalid Permissions"* if the user has permissions, then their query is executed if all goes well with the update query *"New application created successfully"* if for some reason the update query cannot be executed correctly then the user gets the appropriate message.

```
$sql = "INSERT INTO applications(id, date_submitted, startd, endd, Reason,
statuss) VALUES ('$id', '$current_date', '$date1', '$date2', '$reas',
'pending')";
```

root_action.php

Root_action is more or less same as admin_action.php and employee_action.php all these three pages use the same validation process and listing methods with different parameters. In this file we cross reference the users submitted credentials with data we get from the root_acc table on our database and instead of passing the data of **$valid** variable into the **employee_privilages** or **admin_privilages** variables we pass it in **root_privilages** in the users session. Listed for the user are the basic info of all users from the employee and supervisor tables in our database using the below SELECT queries.

```
$sql2 = "SELECT `first_name`, `last_name`, `email`, `username`, `id` FROM
`employee`";
$sql3 = "SELECT `first_name`, `last_name`, `email`, `username`, `id` FROM
`supervisor`";
```

Above the list the user has a button name *"Create New User"* when the user presses the button the root user is taken to the submit_new_user.php.

submit_new_user.php

Here a root user can submit a new user credentials to our database in the way of a form with the following basic fields. First name, Last name, Email, User Type, Username, Password and Re-confirm Password. The user type field of the form gives the user two options either Admin or Employee, all the other fields are text-based inputs. The user submits the data by pressing the Create button, after pressing the button the user is taken to the new_employee.php page.

new_employee.php

Same as **application.php** before the data is inserted using an INSERT INTO query we check the **root_privilages** the same way we check **employee_privilages** on **application.php** and **admin_privilages** on **update_application.php.** If the user does not have permission to submit an INSERT query the user gets an *"Invalid Permissions"* if the user has permissions, then their query is executed if all goes well with the update query *"New user created successfully"* if for some reason the update query cannot be executed correctly then the user gets the appropriate message. Then even before using the INSERT INTO query we check the choice given in the field User Type on the **submit_new_user.php** page, if the choice was Admin then the INSERT INTO query we use is the following.

```
$sql = "INSERT INTO `supervisor`(`first_name`, `last_name`, `email`,
`username`, `passw`, `id`) VALUES
('$fname','$lname','$mail','$nuser','$npass','NULL')";
```

Otherwise the query we use is.

```
$sql = "INSERT INTO `employee`(`first_name`, `last_name`, `email`, `username`,
`passw`, `id`) VALUES ('$fname','$lname','$mail','$nuser','$npass','NULL')";
```

db_connect.php

This php file is required from all the php files that attempt to connect to the applications database in this file we provide the application the needed credentials to contact the database using a **mysqli** object appropriately named mysqli.

Database Tables

Our application uses a database containing four tables listed below is each table and its field and a short description of each fields use.

applications

- id : The id of the employee that submitted the application it has a relation to the employee table seen below.
- application_id : A unique auto incrementing id for each application submitted by users.
- date_submitted : The date the application was submitted, it is used to order the applications by ascending order when viewed.
- startd : The leave start date.
- endd : The leave end date.
- Reason : The reason of the leave.
- statuss : The status of the application.

employee

- first_name : The first name of the employee.
- last_name : The last name of the employee.
- email : The email of the employee.
- username : The username of the employee account.
- passw : The password of the employee account.
- id : The id of the employee account, it is used by the applications table to specify which user made which application.

supervisor

- first_name : The first name of the admin.
- last_name : The last name of the admin.
- email : The email of the admin.
- username : The username of the admin account.
- passw : The password of the admin account.
- id : The id of the admin account.

root_acc

- username : The username of the root account.
- passw : The password of the root account.

Possible Updates

Combining login pages

In a later iteration of the app a much needed update would be the combination of the login and pages into one. That would require the database tables **root_acc, employee, supervisor** to be combined and a new field named **permissions** to be added to the fields, so we could differentiate permissions better. That would make for a much cleaner and streamlined application.

Creating better permission handling

Its clear that this iteration of the code is not hack proof with the handling of permissions and access to the server, with more experience in the handling of permissions and passing of variables through the application pages a more robust version of the application could be achieved.

Addition of CSS

At this point in the development of the application function of the application took priority over the looks of the application, when the functionality we require is achieved then an overhaul of the application using CSS and JavaScript could make the application more appealing to users.